Reconfigurable Security Extensions in Hardware for RISC-V Architecture

Asmit De, Aditya Basu, Swaroop Ghosh and Trent Jaeger
The Pennsylvania State University

Research Description

With the recent proliferation of Internet of Things (IoT) and embedded devices, there is a growing need to develop a security framework to protect such devices. RISC-V is a promising open source architecture that targets low-power embedded devices and SoCs. However, there is a dearth of practical and low-overhead security solutions in the RISC-V architecture. Programs compiled using RISC-V Toolchains are still vulnerable to code injection and code reuse attacks such as buffer-overflow attacks and return-oriented programming (ROP). Previous research have explored software based solutions such as shadow stack, stack canaries, etc. [1-3] to prevent such attacks in x86 architecture, however, they are expensive in terms of power and performance impact and often require significant code/binary instrumentation and compiler modifications. Alternative hardware based solutions have also been explored [4-6], however these too require compiler support and also require new architectural elements and potential modifications to an existing processor core.

Control flow integrity (CFI) is one of the approaches considered to prevent ROP attacks. However, existing software CFI measures require compiler passes to construct a control flow graph (CFG) and then program modifications to validate against the CFG. This validation impacts performance and consumes more energy due to extra memory access. The objective of this research is to develop low overhead reconfigurable security extensions in hardware that ensures integrity of both the backward and forward edge control flow of programs running on a RISC-V core. Our RISC-V coprocessor based solution decouples the security architecture from the RISC-V core architecture, enabling a highly flexible security system design. Such an approach has the potential to be scaled to heterogeneous processor designs such as a Xeon + FPGA core. In such designs, the primary core can be completely unmodified, while the re-configurable FPGA core can be utilized to implement the security architecture. The FPGA also provides the flexibility to change and update the security architecture in demand to new threats, without a complete redesign of the primary computing core.

Demonstration

The security extensions will be demonstrated on a Xilinx Zynq FPGA (Fig. 1). The Zynq FPGA has an ARM core which will be programmed to emulate a RISC-V RocketChip in-order microprocessor with an integrated Rocket Custom Coprocessor (Fig. 2). A front-end server running RISC-V Linux and program binaries will be used communicate with the ARM core to emulate the RISC-V instructions. We will demonstrate the following:

(i) A RISC-V port of the Linux kernel running as a base OS on the RocketChip
(ii) Program vulnerabilities such as buffer overflow and function pointer manipulation
(iii) Updating the Rocket Coprocessor (RoCC) with security extensions
(iv) Backward edge control flow integrity enforced by the RoCC

(v) Forward edge control flow integrity enforced by the RoCC

(vi) Power and performance measures compared to software solutions

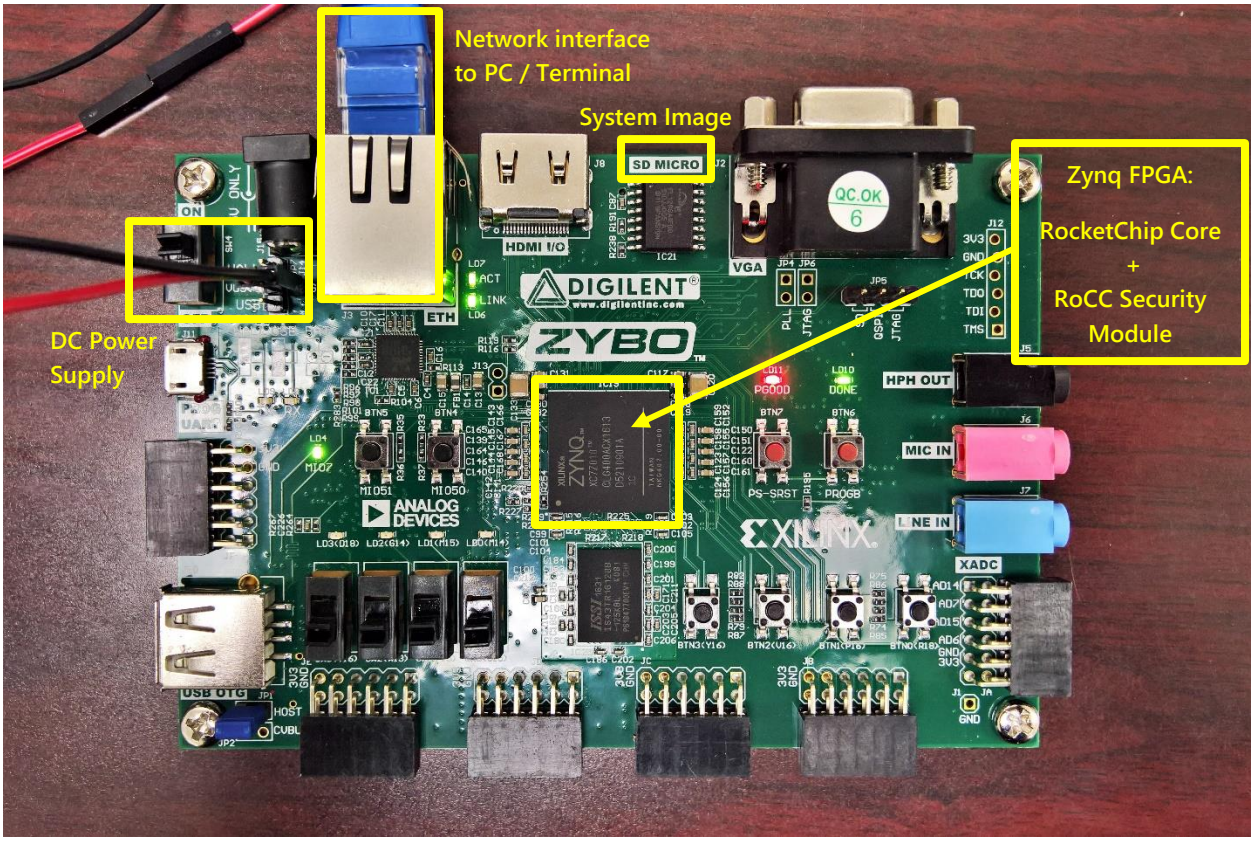

Fig. 1. Experimental setup showing the Zynq FPGA configured for RocketChip Core and RoCC Security Module enforcing control flow integrity. The Linux kernel image and the FPGA configuration bitstream are loaded from the Micro SD card. An SSH connection can be established to the FPGA through the network interface to run program binaries.

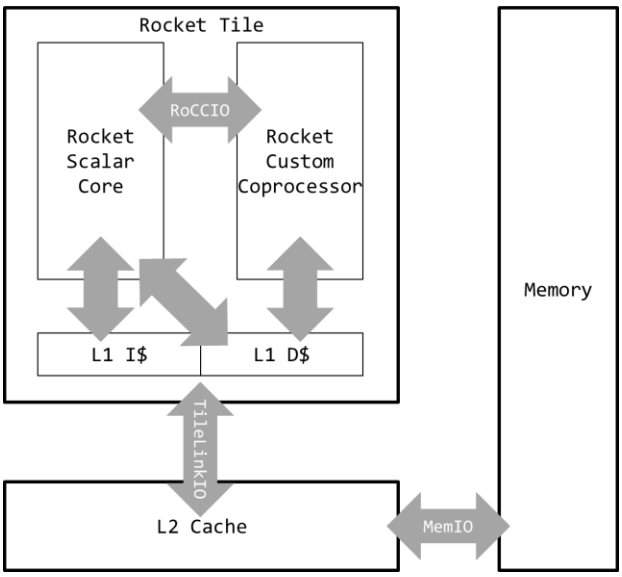

Fig. 2. RISC-V RocketChip architecture

References:

[1] Cowan et al. "StackGuard: automatic adaptive detection and prevention of buffer-overflow attacks." USENIX Security Symposium, 1998.

[2] Pappas et al. "Transparent ROP Exploit Mitigation Using Indirect Branch Tracing." USENIX Security Symposium, 2013.

[3] Cheng et al. "ROPecker: A generic and practical approach for defending against ROP attack." NDSS, 2014.

[4] Davi et al. "HAFIX: Hardware-assisted flow integrity extension." Design Automation Conference, 2015.

[5] Song et al. "HDFI: hardware-assisted data-flow isolation." Security and Privacy (SP), 2016.

[6] Ge et al. "GRIFFIN: Guarding control flows using Intel processor trace." ASPLOS, 2017.